FP03140-IDS (JP07-44401)

Japanese Unexamined Patent Application Publication No. 07-44401

SPECIFICATION <EXCERPT>

[0006]

[Embodiments]

(Embodiment 1) FIG. 1 shows a processor of the present invention which performs process switching with hardware. General register files made up of 256 general registers, namely general registers Rg0 to Rg255, are divided into groups each including 16 general registers, which makes 16 groups, i.e. Grp0 to Grp15. Here, the group division of the present invention is performed without adding hardware to the general register files for the purpose of the division processing. The method for the group division is described later. Management registers are provided in correspondence with the general register groups of the above general registers, and make up management register groups each including management registers the count of which corresponds to the count of groups of the general registers. In each management register, PC* which is either a preset program counter PC value or a program counter PC value at the time of process suspension is stored together with flag information. Here, the execution status flag requires bits to indicate at least the currently-in-execution status and the standby status. To read and write the execution status flag, there are two methods: a method of reading and writing an execution status flag for each register number of the management registers, and a method of reading and writing an individual bit indicating an execution status of each management register. With the method of reading and writing an individual bit indicating an execution status of each management register, the location of the read-out status bit matches the management register number. A specific example is

shown in FIG. 4. FIG. 4 assumes that the flag of a management register 0 corresponding to the general register group Grp. 0 indicates the currently-in-execution status, whereas the flag of a management register 1 corresponding to the general register group Grp. 1 indicates the standby status. The management registers have a flag having a currently-in-execution bit and a standby bit. The general register group Grp. 0 is currently in execution, and thus a 0-bit flag of the management register is set, and the flag having the currently-in-execution bit that is 0 bit is set. Here, to switch the status of the general register group Grp. 1 to the currently-in-execution status, all bits of the currently-in-execution bits are scanned to identify that the flag having the currently-in-execution bit that is 0 bit is set. Then, the currently-in-execution bit in the 0-bit flag of the management register is cleared, the execution status is switched to the standby status, and the flag having the currently-in-execution bit that is 1 bit is set. By doing so, the currently-in-execution bit is switched to the standby bit. Further, being connected with the general registers, an operator ALU which performs logical operations between specified general registers is provided. In addition, a data cache memory and an instruction cache memory are provided which constantly accumulate data of a main memory and read data from the main memory when there is no data to read. These cache memories are a primary cache memory and/or a secondary cache memory. The instruction cache memory reads code instructions, whereas the data cache memory reads and writes data referred to from registers. Further, the data cache memory sends and receives bi-directional data, and reads an address of the program counter PC via an address line. Whereas, the instruction cache memory is connected to a fetch circuit which fetches the received instruction. The fetch circuit is connected to a decoding circuit DCR. The instruction decoded by the decoding circuit DCR is sent, as a control

signal, to the management registers, the general register, a logical circuit such as the operator ALU, and is inputted to a process switching circuit through one of these destinations. Here, the process switching circuit is connected to the decoding circuit DCR when a process switching instruction is generated, and includes a saving block, a selection algorithm block, and a returning block. The saving block, the returning block, and the selection algorithm block are connected to a switching control line connected to a register control line, and control the program counter PC value PC* stored in the management registers and information on the execution status flag. Further, the selection algorithm block generates a register group signal Grp#. In transferring data to the management registers and the general register groups, the register group signal Grp# is used for specifying a corresponding general register group. The management registers and the general register groups are connected to an input data bus and an output data bus. The data outputted from the management registers and the general register groups is transferred to the operator ALU that has been accessed through address specification by the decoding circuit DCR, and is also transferred to the program counter PC. Further, output data from the program counter PC is accessed by the management registers and the general register groups via the input data bus. On the OS, the management registers correspond one-to-one with the general register groups, and in the case where a management register "a" corresponds to the general register group Grp0, for example, the management register "a" is specified when the general register group Grp0 is specified with the register group signal Grp#, and a process stored in the management register "a" is assigned to the general register group Grp0. Data to be recorded in the management registers is not particularly limited, but includes execution data at the time of execution suspension. As a result, according to the present embodiment, in the case where the general

register group Grp0 is specified in association with a single register group signal Grp# at the time of execution suspension, two pieces of data are held, namely, an execution status flag and the program counter PC value PC* stored in the management register "a" corresponding to the general register group Grp. Here, the flag of the management register "a" corresponding to the general register group Grp0 indicates the currently-in-execution status, whereas the flag of a management register "b" corresponding to the general register group Grp1 indicates the standby status.

[0007] The following are descriptions of an initial setting operation and a process switching operation of the processor.

[0008] First described is a method for grouping general registers. Although not particularly shown, a reset signal is inputted to the processor, data of the flags of the management register groups are cleared, and the status of only an activation management register is switched to the execution status. The activation management register stores in advance a PC value for activation, and this value is transferred to the program counter PC to permit reading from a memory to a CPU. With this, an activation program is read to the CPU, necessary data is obtained in the management registers and the general registers in the CPU to allow the OS activation, data on the memory is allocated, a peripheral input-output device is set up, and so on. Then, when the preparation is completed, the OS is activated, and processes are allocated also to the management registers other than the activation management register under the OS management. Here, the processes allocated to the management registers are not particularly limited, and may be processes for processing the OS or for controlling the peripheral input-output device. Further, the processes may be applications activated by the OS.

[0009] The following is a description of procedures of data transfer at the time of process switching, performed by the processor

according to the present embodiment. Here, the following are assumed as described above: the general register files are divided into 16 groups; the general register group Grp0 is in the execution status at the time of process switching; and the process switching is performed so as to switch to the general register group Grp1. A process switching instruction is outputted from the data cache memory and the instruction cache memory. Then, the data cache memory specifies the address of a register via the data bus, and the instruction cache memory fetches the process switching instruction to the processor via the fetch circuit. Then, the instruction fetched by the fetch circuit is transferred to the decoding circuit DCR where the instruction is decoded and operation details of the operator ALU are selected, followed by transfer of data on the operation details to the operator ALU. In addition, the data outputted from the decoding circuit DCR is inputted to the process switching circuit.

[0010] The following is a description of a processing method for use in the process switching circuit. First, the saving block accesses the register control line via the switching control line, and selects the management register "a" that is currently in execution. Then, the saving block holds the program counter value PC* in the program counter PC, and overwrites the program counter value PC* in the management register "a" via the input data bus. The saving block then reads the execution status flag stored in the management register "a" into the switching control line via the register control line, and switches the execution status flag to the standby status flag. Next, the selection algorithm block selects a process to be executed next from the standby status flag, and generates the register group signal Grp# to specify a general register group corresponding to the selected process. Then, while a latch circuit holds the register group signal Grp# generated by the selection algorithm block, the general register group Grp1 is selected, and by doing so, the management register "b" is specified. Moreover, the

returning block specifies the management register "b" via the register control line accessed through the switching control line, reads the program counter value PC* stored in the management register "b" via the output data bus, and sets the program counter PC to the program counter value PC*. Then, the management register "b" is identified based on the register group number read by the selection algorithm block using a bit number, and the standby status flag stored in the management register "b" is switched to the bit of the execution status flag during the execution of the execution status flag. As the instruction cache memory reads an initial instruction of the selected process into the fetch circuit, the process switching operation finishes, and the next process is executed. All the processes do not need to be managed only by the process switching circuit and the management registers, but may also be managed by the main memory. In fact, the number of processes managed by the OS is so great that all the processes cannot be managed by the management registers and so on in the processor CPU. Thus, when utilizing the present invention in the OS, the most-frequently-used standby process is stored in the management registers, and management register data regarding a process, the execution of which has finished or has been suspended is stored in the main memory. Here, the processing of the OS includes: saving data from the management registers to the main memory; copying data of a standby process in the main memory to the management registers; and selecting a process to be copied to the management registers. When there is data in the memory, necessary data is transferred from the data cache memory to the management register groups through program control or automatic control on data which is less frequently used among the management registers and the general register groups Grp0 to 15. It is based on the premise that the processor CPU does not execute the next instruction until this switching operation finishes. This switching

method can be implemented using both software and hardware. In the case of software, at least one management register is occupied by activating the switching process. Conventionally, occupying a general register file in one process has resulted in consumption of several tens of cycles in the saving operation. Thus, it is desirable to use hardware to achieve high speed using this method. However, in this case, it is not that hardware is added to the general register file for the purpose of the division processing, but it is for converting a register address assigned to each process into an address of the general register file. Although the present embodiment has described the ALU as the operator, it is possible to use Floating Point Processing Unit instead to further reduce the burden on the software of the whole system.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-044401
(43)Date of publication of application : 14.02.1995

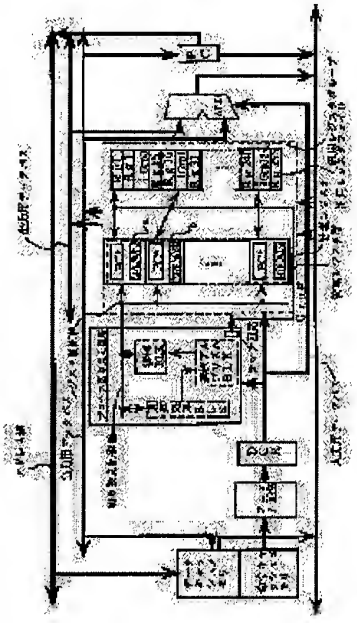
(51)Int.Cl. G06F 9/46

(21)Application number : 05-192157
(22)Date of filing : 03.08.1993

(71)Applicant : HITACHI LTD
(72)Inventor : IKETANI TOYOHITO

(54) LOGIC INTEGRATED CIRCUIT AND ITS DATA PROCESSING SYSTEM

(57)Abstract:
PURPOSE: To control efficiently a multi-processor comprising plural CPUs.
CONSTITUTION: An area of a general-purpose register used by one process is limited so that data for the general-purpose register are not saved in an external memory at the changeover of the execution process in an RISC processor and plural processes are reserved by dividing the area of the general-purpose register. In the division of the general-purpose register, a management register is provided, which reserves information of the division of the general-purpose register and information for the process changeover and a register address when an instruction is read is converted into an address of the general-purpose register when the instruction is decoded, As the division information for the case, the address of the general-purpose register to each process is referenced by the management register. Furthermore, two program counters are provided in one CPU, the one program counter is used for the execution process and the other is used for an auxiliary use and set in the execution standby state and an optional process in the standby is allocated to the divided general-purpose register.



(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平7－44401

(43)公開日　平成７年(1995)２月14日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
G06F 9/46	313 C	8120－5B		

審査請求　未請求　請求項の数6　ＯＬ　（全 17 頁）

(21)出願番号　特願平5－192157

(22)出願日　平成５年(1993)８月３日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　池谷　豊人
東京都青梅市今井2326番地　株式会社日立製作所デバイス開発センタ内

(74)代理人　弁理士　小川　勝男

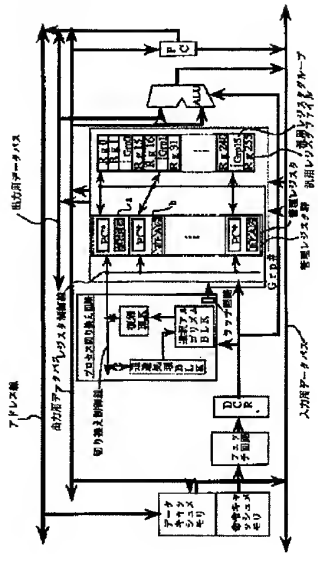
(54)【発明の名称】　論理集積回路およびそのデータ処理システム

(57)【要約】　（修正有）

【目的】複数のＣＰＵで構成されるようなマルチプロセッサを効率的に制御する。

【構成】ＲＩＳＣプロセッサにおいて、実行プロセスの切り換え時に汎用レジスタのデータを外部メモリに退避しなくても良いように、１つのプロセスで使用できる汎用レジスタの領域を限定し、複数のプロセスを上記汎用レジスタ内の分割により確保できるようにする。汎用レジスタの分割は、汎用レジスタ内の分割に対する情報及びプロセス切り換えに対する情報を確保している管理レジスタを設け、命令読み込み時のレジスタアドレスを命令解読時に汎用レジスタアドレスに変換する。そのときの分割情報として、汎用レジスタの各プロセスに対する値を管理レジスタから参照する。さらに、１ＣＰＵ内に２つのプログラムカウンタを設け、１つを実行用のプロセス、その他のプログラムカウンタを補助用とし、実行待ち状態として分割レジスタに待機中の任意のプロセスに割り当てる。

【図１】



1

【特許請求の範囲】

【請求項１】複数の汎用レジスタから成る汎用レジスタグループによって構成される汎用レジスタファイルと、上記汎用レジスタ夫々に割り当てられるプロセスの情報と該プロセスの切り換えに関する情報を格納する複数の管理レジスタから成る管理レジスタ群と、メモリとを１チップ内に有する論理集積回路であって、上記管理レジスタは上記汎用レジスタファイル内の汎用レジスタおよび上記メモリとの間で読み出し動作及び書き込み動作を行なわせる命令を保持しているとともに、上記管理レジスタはプロセスごとに分割される機能を有するものであって上記汎用レジスタは実行プロセスごとに上記管理レジスタによって割当てられ、実行プロセスが異なった場合かつ命令入力時に同一の汎用レジスタを指定することにより夫々の実行プロセスに割り当てられた汎用レジスタに指定され、命令入力時のレジスタ指定から上記管理レジスタ内に格納されたデータを参照して、夫々のプロセスごとに上記汎用レジスタの使用範囲を限定することを特徴とする論理集積回路。

【請求項２】上記汎用レジスタファイル内の夫々の汎用レジスタに、実行状態のプロセスと、実行待ち状態のプロセスに関する情報を格納することを特徴とする特許請求の範囲第１項記載の論理集積回路。

【請求項３】上記論理集積回路は、プロセッサを複数所有し、上記汎用レジスタにて複数のプロセスを同時に実行することを特徴とする特許請求の範囲第１項記載の論理集積回路。

【請求項４】上記論理集積回路は、実行プロセスの命令コードを示すプログラムカウンタと、実行待ち状態にある実行プロセス以外のプロセスの命令コードを示すプログラムカウンタと、該命令コードを検出するための回路とを有し、通常処理では上記実行プロセスのプログラムカウンタを使用するとともに、命令を読み込む前に上記命令コード検出回路によってパイプラインで実行されない命令コードを検出し、上記実行待ち状態のプロセスの命令コードを示すプログラムカウンタに切り換えるとともに別のプロセスの命令コードを代わりに挿入して、命令入力時にパイプラインで実行可能な命令コードに置き換えるプログラムカウンタ切り換え回路を備えてなることを特徴とする特許請求の範囲第１項記載の論理集積回路。

【請求項５】上記プロセスの切り換えを、管理レジスタを選択しプログラムカウンタにその値を保持するとともに上記管理レジスタに上書きし実行状態フラグを実行待ち状態フラグに切り換える機能と、上記実行待ち状態フラグから次実行プロセスを選択しそのプロセスと対応した汎用レジスタグループを指定する選択アルゴリズムに従ってデータを授受する機能と、次に実行する管理レジスタを指定しそこに格納されたプログラムカウンタの値を読み出すとともに上記プログラムカウンタにセットするデータ復帰機能とを有するプロセス切り換え回路により行なうとともに、上記選択アルゴリズムにおいてはビット番号により読み出された汎用レジスタグループから次実行プロセスの情報を有する管理レジスタを割り出し上記管理レジスタに格納された待機状態のフラグを実行状態フラグの実行中に同一ビットに切り換えることを特徴とする特許請求の範囲第１項記載の論理集積回路。

2

【請求項６】システムバスにより、複数のプロセッサ，メモリマネージメントユニット，メインメモリ，グラフィックアクセラレータ，入出力装置，キャッシュメモリが接続されているとともに、上記グラフィックアクセラレータとＣＲＴ、上記入出力装置とネットワーク，キーボード，ハードディスクとが接続され、上記メモリマネージメントユニットにて階層化したメモリと実メモリとを変換し、上記グラフィックアクセラレータにて高速描画を命令により読み込み実行し、上記グラフィックアクセラレータにて演算結果をＶＲＡＭへドットデータ出力として書き込み上記ＣＲＴに出力，表示を行ない、上記入出力装置にて上記プロセッサからの命令にしたがって外部とのデータの入出力を行なうとともにＤＭＡ転送の際にはメモリー入出力装置間のデータ転送を行ない、上記入出力装置にて上記キーボードから入力された外部からの命令を取り込むとともに、上記ネットワークにて他のシステムとの通信を行ない、上記ハードディスクにて上記プロセッサから発生されたハードディスクドライバに対して該ハードディスクの内容を取り出す量とそのメインメモリへの格納場所を指定するデータ処理システムであって、複数の汎用レジスタから成る汎用レジスタグループによって構成される汎用レジスタファイルと、上記汎用レジスタ夫々に割り当てられるプロセスの情報と該プロセスの切り換えに関する情報を格納する複数の管理レジスタから成る管理レジスタ群と、メモリとを１チップ内に有する論理集積回路であって、上記管理レジスタは上記汎用レジスタファイル内の汎用レジスタおよび上記メモリとの間で読み出し動作及び書き込み動作を行なわせる情報を保持しているとともに、上記管理レジスタはプロセスごとに分割される機能を有するものであって上記汎用レジスタは実行プロセスごとに上記管理レジスタによって割当てられ、実行プロセスが異なった場合かつ命令入力時に同一の汎用レジスタを指定することにより夫々の実行プロセスに割り当てられた汎用レジスタに指定され、命令入力時のレジスタ指定から上記管理レジスタ内に格納されたデータを参照して、夫々のプロセスごとに上記汎用レジスタの使用範囲を限定するものであることを特徴とするデータ処理システム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は論理集積回路における演算回路に適用して有効な技術であり、特に大容量の汎用レジスタを有するマイクロプロセッサおよびデータ処理

システムに利用して有効な技術に関する。

【0002】

【従来の技術】近年、ＲＩＳＣプロセッサの発展にともないマイクロプロセッサの性能が飛躍的に向上してきている。一般に、このＲＩＳＣプロセッサはパイプライン技術を駆使し動作を高速化するために、各パイプラインを１マシンサイクルで実行できるように単純な命令だけで構成されているマイクロプロセッサである。そして、図２に示すようにこのＲＩＳＣプロセッサにおいては、動作周波数が高く、１つの命令が単純であるため、大容量かつ大量の命令を必要とし、ＣＰＵ内蔵の１次キャッシュメモリと２次キャッシュメモリとが設けられている。そして、実行頻度の少ない汎用レジスタのデータをメインメモリへも退避させている。このため、パイプラインの流れを乱し、高速化を妨げるような上記１次キャッシュメモリ，２次キャッシュメモリ，メインメモリのそれぞれとＣＰＵ内汎用レジスタ間の演算、上記２次キャッシュメモリと上記１次キャッシュメモリ間の演算を避け、上記汎用レジスタと他の上記汎用レジスタ間の演算のみが行われるようにされている。また同様な理由から、このＲＩＳＣプロセッサにおいてはＣＩＳＣプロセッサにみられる、次データも読み出せるように次アドレスをも決定するオートインクリメント・デクリメントに代表されるようなパイプラインの前後関係に依存する複雑なアドレッシングモードは採用されていない。図３に一般的なＲＩＳＣプロセッサのパイプライン処理の概念図を示す。このＲＩＳＣプロセッサにおいては、命令読み出し、命令解読、汎用レジスタ指定、ＡＬＵ演算、ライトバックのような一連の動作がパイプラインで行われることによって命令の実行が行われている。そして、メモリをアクセスするときには演算器ＡＬＵ，ライトバックの間にメモリアクセスを短縮化するために１ステージが追加される。しかし、ＣＩＳＣプロセッサにおいてはプログラマによるアセンブラレベルでのプログラミング工数を低減するために、複雑な命令，アドレスを取り込み、パイプラインの乱れによる処理速度の低下を軽視していた。また、ＣＩＳＣプロセッサではパイプラインの乱れによって数サイクル余分に必要となるために、命令実行が遅くなるという問題点がある。一方、上記ＲＩＳＣプロセッサでは基準とする汎用レジスタとインデクスとの演算を行うことによって汎用レジスタを指定する、汎用レジスタ間の演算だけによるインデクス修飾アドレッシングモードを採用して１サイクルでアドレス計算できるため、ＲＩＳＣプロセッサでは１サイクルで命令の実行ができる。しかしながら、上記ＲＩＳＣプロセッサ，ＣＩＳＣプロセッサを問わず既存のプロセッサにおいては、１つの実行プロセスで上記汎用レジスタを専有させる方式を採っている。このため、外部メモリへ退避させたデータの上記外部メモリから上記データの汎用レジスタへのデータ転送を軽減するために、大量の汎用レジスタを設けている。そして、最も使用頻度の多い変数のみを上記汎用レジスタに残すようにして、コンパイラで上記汎用レジスタに割り当てられる変数を最適化している。このように、従来のＲＩＳＣプロセッサにおいては、ハードウエア技術の改善の他にソフトウェア技術の向上に負うところが大きかった。しかし、たいていのコンピュータシステム，ワークステーション等は、複数のソフトウェアを同時に実行するために時分割でプロセスを切り換えるマルチタスクシステムをサポートしているため、一定時間内に複数の実行プロセスを切り換えなければならない。このとき、大量の汎用レジスタを１つの実行プロセスで占有していると、プロセス切り換え時にオペレーティングシステム（以下ＯＳと記す）上で上記大量の汎用レジスタに格納されたデータを外部メモリに退避しなければならなくなる。このことにより、外部メモリ－汎用レジスタ間のアドレッシングによるデータ退避のため、実行中のアプリケーションソフトウェアからは無用な遅延時間が生じる。また、これに対し、ほとんどシングルチッププロセッサ構成を前提としている既存のコンピュータシステム，ワークステーション等はプロセス切り換え時にデータを退避させることが原因となってシステム全体のスループットを犠牲にしない程度の退避時間で済むように、汎用レジスタの数を制限することで対処していた。ところが、マルチチッププロセッサで構成されたコンピュータシステムにおいては、実行プロセス単位で各プロセッサを割り当てる必要性が生じ、プロセス切り換え手段が複雑になるとともに、さらにプロセス切り換え時の遅延時間が大きくなり同様の問題が生じている。このため、従来のＯＳではＲＩＳＣプロセッサは複数のプロセスの選択手段を決めるだけだったが、マルチプロセッサ用のＯＳではそれに加えてプロセッサの選択を行うことが必要となっている。また、分散ＯＳにおいてはさらにスレッドのようなサブルーチンレベルのさらに細かいプロセスにより、ソフトウェアが構成されているため、頻繁にプロセス切り換えを行わなければならない。このため、前述したようなＲＩＳＣプロセッサのアーキテクチャをマルチプロセッサに適用した場合に、すべてのプログラムをプロセッサごとに振り分ける必要性が生じている。このことにより、頻繁なプロセス切り換えが行われ、パイプライン処理に支障が生じ、汎用レジスタの外部メモリへのデータの退避がシステム全体にとって多大な負担となってしまう。

【0003】

【発明が解決しようとする課題】本発明は、次世代のＯＳに対応できる高速なプロセス管理及びレジスタ管理機能を有するマイクロプロセッサを提供し、１ＬＳＩ内に複数のＣＰＵで構成されるようなマルチプロセッサを効率的に制御する管理機能を有するマイクロプロセッサを提供することを目的とする。

【0004】

【課題を解決するための手段】ＲＩＳＣプロセッサにおいて、実行プロセスの切り換え時に汎用レジスタのデータを外部メモリに退避しなくても良いように、１つのプロセスで使用できる汎用レジスタの領域を限定し、複数のプロセスを上記汎用レジスタ内の分割により確保できるようにする。そして、上記汎用レジスタ内の分割に対する情報及びプロセス切り換えに対する情報を確保している管理レジスタを設け、命令読み込み時のレジスタグループ内のレジスタアドレスを命令解読時に汎用レジスタアドレスに変換する。そのときの分割情報として、上記汎用レジスタの各プロセスに対するプログラムカウンタの値を上記管理レジスタから参照する。さらに、１ＣＰＵ内に２つのプログラムカウンタを設け、１つを実行用のプロセス、その他のプログラムカウンタを補助用とし、実行待ち状態として分割レジスタに待機中の任意のプロセスに割り当てる。

【０００５】

【作用】上記レジスタ分割機能を有するマイクロプロセッサは、１ＣＰＵ内に複数のプロセス情報を有することができ、１～数サイクル以内でプロセスを切り換えることが可能となる。このことにより、実行プロセスの切り換えに対する汎用レジスタの外部メモリへ退避するデータ量を最小限に抑えることができ、切り換え時間を大幅に短縮することができる。また他のプロセスと並行処理することにより、実行中の命令コードの中でディレイスロットが原因で実行できない命令コードを検出したときに、プログラムカウンタを待機中のプロセスに切り換え、実行可能な命令コードに置き換えることによりパイプラインの乱れを軽減することができる。

【０００６】

【実施例】

（実施例１）図１にハードウエアでプロセス切り換えを行なう本発明のプロセッサを示す。汎用レジスタＲｇ０～Ｒｇ２５５のように２５６個の汎用レジスタによって構成される汎用レジスタファイルをそれぞれ１６個ずつの汎用レジスタでグルーピングし、Ｇｒｐ０～Ｇｒｐ１５まで１６個のグループに分割する。ここで、本発明においては、上記汎用レジスタファイル自身に分割のためのハードウェアは付加することなくグループ分割が行なわれるものであるがグルーピング方法については後に説明する。そして、上記汎用レジスタにおけるそれぞれの汎用レジスタグループと対応して管理レジスタが構成されるとともに、この汎用レジスタのグループ数と対応した数の上記管理レジスタによって管理レジスタ群が構成される。また、上記それぞれの管理レジスタには予め設定されたプログラムカウンタＰＣの値またはプロセス中断時のプログラムカウンタＰＣの値ＰＣ＊が格納されていると共にフラグの情報が格納されている。ここで、上記実行状態フラグは、少なくとも実行中と実行待ちを示すビットを必要とする。この実行状態フラグを読み書きするときには、それぞれの管理レジスタのレジスタ番号ごとに、ある実行状態フラグを読み書きする方法と、上記それぞれの管理レジスタ内の実行状態を示す個別ビットを読み書きする方法がある。上記それぞれの管理レジスタ内の実行状態を示す個別ビットを読み書きする方法においては、読み出した状態ビットの位置と管理レジスタの番号が一致しているものである。この具体的な例を図４に示す。図４では、汎用レジスタグループＧｒｐ．０と対応した管理レジスタ０のフラグが実行状態で、汎用レジスタグループＧｒｐ．１と対応した管理レジスタ１のフラグが実行待ち状態であるとして説明する。上記管理レジスタは実行中ビットと実行待ち状態ビットを有するフラグを有しており、汎用レジスタグループＧｒｐ．０が実行状態であるために０ビットの管理レジスタのフラグがたっており、上記０ビットの実行中ビットのフラグがたっている。ここで、上記汎用レジスタグループＧｒｐ．１を実行状態に切り換えるために、実行状態ビットの全ビットをスキャンし、０ビットの実行中ビットのフラグがたっていることを割り出す。そして、管理レジスタの０ビットのフラグにおける実行中ビットをクリアして、実行待ち状態に切り換え、１ビットの実行中ビットのフラグをたてる。このことによって、実行中ビットと実行待ちビットを切り換えるものである。さらに、上記汎用レジスタと接続されて、指定された汎用レジスタ間の論理演算を行うための演算器ＡＬＵが構成される。そして、常にメインメモリのデータを蓄積し、読み出すべきデータがないときにメインメモリから読み出すデータキャッシュメモリ，命令キャッシュメモリが設けられている。そして、これらのキャッシュメモリは１次キャッシュメモリ及びまたは２次キャッシュメモリであって、上記命令キャッシュメモリはコード命令を読み、上記データキャッシュメモリはレジスタから参照するデータを読み書きする。さらに、上記データキャッシュメモリは双方向データの入出力を行うとともに、アドレス線を介して上記プログラムカウンタＰＣのアドレスを読み込む。一方、上記命令キャッシュメモリは、上記入力された命令をフェッチするフェッチ回路と接続され、さらに上記フェッチ回路はデコード回路ＤＣＲと接続される。そして、上記デコード回路ＤＣＲにより解読された命令は制御信号として、管理レジスタ，汎用レジスタ，演算器ＡＬＵ等の論理回路にそれぞれ接続され、そのうちの１本がプロセス切り換え回路に取り込まれる。ここで、このプロセス切り換え回路は、プロセス切り換え命令が発生したときに上記デコード回路ＤＣＲと接続されるもので、退避処理ブロック，選択アルゴリズムブロック，復帰ブロックから構成されるものである。そして、上記退避処理ブロック，復帰ブロック及び選択アルゴリズムブロックはレジスタ制御線と接続された切り換え制御線に接続され、上記管理レジスタに格納された上記プログラムカウンタＰＣの値ＰＣ＊及び実行状態

フラグの情報を制御するものである。また、上記選択アルゴリズムブロックによりレジスタグループ信号Ｇｒｐ＃が発生され、上記管理レジスタ及び上記汎用レジスタグループにデータ転送が行われるが、このレジスタグループ信号Ｇｒｐ＃は割り当てた汎用レジスタグループを指定するためのものである。さらに、上記管理レジスタ及び上記汎用レジスタグループは入力用データバス及び出力用データバスと接続される。そして、上記管理レジスタ及び上記汎用レジスタグループから出力されたデータは、上記デコード回路ＤＣＲによりアドレス指定されアクセスされた演算器ＡＬＵに転送されるとともに上記プログラムカウンタＰＣにも転送される。さらに、このプログラムカウンタＰＣからの出力データは上記入力用データバスを介して上記管理レジスタ及び上記汎用レジスタグループとアクセスされる。また、ＯＳ上で上記管理レジスタは上記汎用レジスタグループと１対１で対応させており、例えば管理レジスタａと汎用レジスタグループＧｒｐ０が対応していたとすれば、レジスタグループ信号Ｇｒｐ＃で上記汎用レジスタグループＧｒｐ０を指定した場合には、上記管理レジスタａが指定され、上記管理レジスタａに格納されたプロセスが上記汎用レジスタグループＧｒｐ０に割り当てられる。また、上記管理レジスタに記録するべきデータは特に限定しないが、実行を中断したときの実行データを含むものとする。このことにより、本実施例では１つのレジスタグループ信号Ｇｒｐ＃に対して、実行を中断したときに汎用レジスタグループＧｒｐ０が指定されていたとすれば、上記汎用レジスタグループＧｒｐに対応する上記管理レジスタａに格納された上記プログラムカウンタＰＣの値ＰＣ＊と実行状態のフラグの２つのデータが保持されていることになる。ここで、上記汎用レジスタグループＧｒｐ０に対応する上記管理レジスタａのフラグは実行中の実行状態フラグであり、汎用レジスタグループＧｒｐ１に対応する管理レジスタｂのフラグは待機状態の実行状態フラグを示している。

【０００７】次にこのプロセッサの初期設定動作及びプロセス切り換え動作について説明する。

【０００８】まず、汎用レジスタのグルーピングの方法について説明する。特に図示しないが、プロセッサにリセット信号を入力し、管理レジスタ群のフラグのデータをクリアし、起動用の管理レジスタのみを実行状態にする。上記起動用の管理レジスタにはあらかじめ起動時のＰＣ値が格納されており、この値をプログラムカウンタＰＣに転送して、メモリからＣＰＵへの読み込みを許可する。これによって、起動用プログラムが上記ＣＰＵに読み込まれ、さらにＯＳが起動できるように上記ＣＰＵ内の管理レジスタ及び汎用レジスタ上に必要なデータを確保し、上記メモリ上のデータの割り当て、周辺入出力装置のセットアップ等を行なう。そして、準備が完了したら上記ＯＳを立ち上げ、起動用以外の管理レジスタも上記ＯＳ管理の元でプロセスが割り当てられる。ここで、上記管理レジスタに割り当てられるプロセスは、ＯＳ自身を処理しても、入出力装置周辺を制御しても、ＯＳが立ち上げたアプリケーションであっても特に限定されない。

【０００９】次に、プロセス切り換え時の本実施例のプロセッサのデータ転送手順を説明する。ここでは、上述したように汎用レジスタファイルを１６個のグループに分割し、プロセス切り換え時に汎用レジスタグループＧｒｐ０が実行状態であったとし、汎用レジスタグループＧｒｐ１にプロセス切り換えを行なうとして説明する。プロセス切り換え命令がデータキャッシュメモリ及び命令キャッシュメモリから出力される。そして、上記データキャッシュメモリはデータバスを介してレジスタのアドレス指定を行い、上記命令キャッシュメモリはフェッチ回路を介してプロセッサ内部へプロセス切り換え命令をフェッチする。そして、上記フェッチ回路によってフェッチされた命令はデコード回路ＤＣＲに転送され、命令解読が行われるとともに演算器ＡＬＵの演算内容を選択し、上記演算器ＡＬＵへその演算内容のデータを転送する。さらに、上記デコード回路ＤＣＲから出力されたデータはプロセス切り換え回路に入力される。

【００１０】つぎにこのプロセス切り換え回路における処理方法について以下に説明する。まず、退避ブロックにおいては、切り換え制御線を介してレジスタ制御線とアクセスし、実行中の管理レジスタａを選択する。そして、プログラムカウンタＰＣに上記プログラムカウンタの値ＰＣ＊を保持し、さらに入力用データバスを介して上記プログラムカウンタの値ＰＣ＊を上記管理レジスタａに上書きする。そして、上記管理レジスタａに格納された実行状態フラグを上記レジスタ制御線を介して上記切り換え制御線上に読み出し、上記実行状態フラグを待機状態に切り換える。次に選択アルゴリズムブロックにおいては、実行待ち状態フラグから次に実行するプロセスを選択し、そのプロセスと対応した汎用レジスタグループを指定するためにレジスタグループ信号Ｇｒｐ＃を発生する。そして、ラッチ回路にて上記選択アルゴリズムブロックから発生したレジスタグループ信号Ｇｒｐ＃を保持しつつ、汎用レジスタグループＧｒｐ１を選択し、このことにより上記管理レジスタｂが指定される。さらに、復帰ブロックにおいて、上記切り換え制御線を介してアクセスされるレジスタ制御線により、上記管理レジスタｂを指定し、この管理レジスタｂに格納されたプログラムカウンタの値ＰＣ＊を上記出力用データバスを介して読み出すとともに、このプログラムカウンタの値ＰＣ＊を上記プログラムカウンタＰＣにセットする。そして、上記選択アルゴリズムブロックにてビット番号により読み出されたレジスタグループ番号から上記管理レジスタｂを割り出し、上記管理レジスタｂに格納された待機状態のフラグを実行状態フラグの実行中にそれと

同一ビットに切り換える。そして、選択されたプロセスの最初の命令が上記命令キャッシュメモリよりフェッチ回路に読み込まれることによって、プロセス切り換え動作が終了し、次のプロセスの実行が行われる。また、全てのプロセスは、上記プロセス切り換え回路及び管理レジスタのみで管理される必要はなく、メインメモリで管理することもできる。実際、ＯＳで管理しているプロセスは莫大であり、プロセッサＣＰＵ内の管理レジスタ等で全てのプロセスを管理することはできない。よって、本発明をＯＳ上で利用する場合は、最も使用頻度の多い実行待ち状態のプロセスを管理レジスタに保管しておき、実行を終了もしくは中断したプロセスは管理レジスタデータをメインメモリへ保存する。ここで、ＯＳが行なう処理は、上記管理レジスタから上記メインメモリへ退避する処理と、上記メインメモリ上にある実行待ち状態のプロセスのデータを上記管理レジスタへコピーをする処理と、上記管理レジスタへコピーすべきプロセスを選択する処理を含む。ところで、メモリ上にデータがあるときは、上記管理レジスタ及び汎用レジスタグループＧｒｐ０～１５のうちで使用頻度の少ないデータをプログラム制御あるいは自動制御により上記データキャッシュメモリから管理レジスタ群へ必要なデータを転送する。この切り換え作業が終了するまでプロセッサＣＰＵは次の命令を実行しないことを前提とする。この切り換え方法はソフトウェアとハードウェア両方による実現が可能であるが、ソフトウェアの場合は切り換えプロセスを起動することにより少なくとも１つの管理レジスタを占有する。また、従来は１つのプロセスで汎用レジスタファイルを占有することによって、退避作業に数１０サイクルが消費されていたので、本方式を用いて高速化を図るためにはハードウェアで組み込む方が望ましい。しかし、この場合、汎用レジスタファイル自身に分割のためのハードウェアを付加するものではなくプロセス単位に割り当てたレジスタアドレスを汎用レジスタファイルのアドレスに変換するものである。ここで、本実施例では演算器としてＡＬＵを記載したが、かわりにＦｌｏａｔｉｎｇ Ｐｏｉｎｔ　Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔを使用することによりさらにシステム全体のソフトウェアの負担が軽減できる。

【００１１】（実施例２）管理レジスタの数と汎用レジスタグループの数が一致していない場合でも、上記管理レジスタに任意に汎用レジスタグループを選択させる機能をもたせたプロセッサを図５に示す。本実施例においては、実施例１における管理レジスタ群のそれぞれの管理レジスタに汎用レジスタのグループ番号に関するデータも格納させたものである。本実施例においては、管理レジスタａ，管理レジスタｂに格納された汎用レジスタグループ番号に関するデータｇｒｐ＊は、それぞれ汎用レジスタグループＧｒｐ０，Ｇｒｐ２のデータであるものとして以下に説明する。まず、実施例１と同様にしてＧｒｐ０，Ｇｒｐ２のように汎用レジスタのグルーピングを行ない、ＯＳ上でプログラムカウンタＰＣの値ＰＣ＊をそれぞれの上記管理レジスタにそれぞれ設定するとともに、実行状態フラグデータ，グループ番号データｇｒｐ＊を格納させる。そして、起動アドレス入力で上記管理レジスタ群の管理レジスタ指定を行ない、上記汎用レジスタグループと対応したプロセスを上記管理レジスタと対応した上記汎用レジスタグループへ転送し実行する。

【００１２】次に本実施例によるプロセッサのプロセスの切り換え方法について説明する。ここでは、上述したように汎用レジスタファイルを任意の数のグループに分割し、グループ番号データｇｒｐ＊として汎用レジスタグループＧｒｐ０のデータが格納された管理レジスタａが実行状態であって、グループ番号データｇｒｐ＊として汎用レジスタグループＧｒｐ２のデータが格納された管理レジスタｂへプロセスを切り換えるとして説明する。命令キャッシュメモリから切り換え命令が発生し、上記命令キャッシュメモリはフェッチ回路を介してプロセッサ内部へ切り換え命令をフェッチする。そして、上記フェッチ回路によってフェッチされた命令はデコード回路ＤＣＲに転送され、データはプロセス切り換え回路に入力される。　つぎに、このプロセス切り換え回路におけるデータ転送方法について以下に詳細に説明する。まず、退避処理ブロックにおいては、切り換え制御線を介してレジスタ制御線とアクセスし、実行中の管理レジスタａを選択する。そして、プログラムカウンタの値ＰＣ＊をプログラムカウンタＰＣに保持し、さらに入力用データバスを介して上記プログラムカウンタの値ＰＣ＊を上記管理レジスタａに上書きする。そして、上記管理レジスタａに格納された実行状態フラグをレジスタ制御線を介して切り換え制御線上に読み出し、上記実行状態フラグを待機状態に切り換える。次に選択アルゴリズムブロックにおいて次に実行するプロセスを選択し、そのデータを復帰ブロックに転送する。そして、この復帰ブロックにおいては、管理レジスタｂを上記切り換え制御線と接続された上記レジスタ制御線を介してアクセスすることにより指定する。そして、上記管理レジスタｂに格納されたグループ番号データｇｒｐ＊の汎用レジスタグループＧｒｐ２のデータを上記レジスタ制御線を介して上記切り換え制御線上に読み出す。そして、上記管理レジスタｂから読み出されたグループ番号データｇｒｐ＊としての汎用レジスタグループデータとしてのＧｒｐ２のデータを選択するためにレジスタグループ信号Ｇｒｐ＃を発生するとともにラッチ回路にてその信号を保持しつつ、汎用レジスタグループＧｒｐ２を指定する。このとき、さらにこの管理レジスタｂに格納されたプログラムカウンタの値ＰＣ＊を上記出力用データバスを介して読み出すとともに、このプログラムカウンタの値ＰＣ＊を上記プログラムカウンタＰＣにセットする。そし

て、上記選択アルゴリズムブロックにてビット番号により読み出されたレジスタグループ番号から上記管理レジスタｂを割り出し、上記管理レジスタｂに格納された待機状態のフラグを実行状態フラグの実行中にそれと同一ビットに切り換える。そして、選択されたプロセスの最初の命令が上記命令キャッシュメモリよりフェッチ回路に読み込まれることによって、プロセス切り換え動作が終了し、次のプロセスの実行が行われる。ここで、実施例１と同様に、全てのプロセスは、上記プロセス切り換え回路及び管理レジスタのみで管理される必要はなく、メインメモリで管理することもできる。実際、ＯＳで管理しているプロセスは莫大であり、プロセッサＣＰＵ内の管理レジスタ等で全てのプロセスを管理することはできない。よって、本発明をＯＳ上で利用する場合は、最も使用頻度の多い実行待ち状態のプロセスを管理レジスタに保管しておき、実行を終了もしくは中断したプロセスは管理レジスタデータをメインメモリへ保存する。ここで、ＯＳが行なう処理は、上記管理レジスタから上記メインメモリへ退避する処理と、上記メインメモリ上にある実行待ち状態のプロセスのデータを上記管理レジスタへコピーをする処理と、上記管理レジスタへコピーすべきプロセスを選択する処理を含む。また、従来は１つのプロセスで汎用レジスタファイルを占有することによって数１０サイクルが消費されていたので、本方式を用いて高速化を図るためにはハードウェアで組み込む方が望ましい。しかし、この場合、汎用レジスタファイル自身に分割のためのハードウェアを付加するものではなくプロセス切り換えを行うものである。また、ここで本実施例では演算器としてＡＬＵを記載したが、かわりにＦＰＵを使用することによりさらにソフトウェアの負担が軽減できる。さらに、汎用レジスタグループの数の方が管理レジスタの数よりも多く、上記汎用レジスタグループに対応する数の管理レジスタが不足した場合にはソフトウェアによって対応する管理レジスタ群を分割してメモリ上に新たな管理レジスタをつくることによって対処する。この実施例においては、ソフトウェアで汎用レジスタ上にあるプロセスデータの参照を行なうことができるため、実施例１と比較してユーザの使い勝手が非常に良くなる。

【００１３】（実施例３）図６に本発明を応用して、フェッチ回路，デコード回路，演算器を複数設け、上記演算器と同数のプログラムカウンタを設けることにより、１つのＬＳＩの中に形成したマルチプロセッサを示す。本実施例では、１つの管理レジスタ群と汎用レジスタファイルが全てのプロセッサごとに共通にアクセスできるように設けられ、上記管理レジスタ群と上記汎用レジスタファイルを分割してプロセッサ別に対応させて命令を受けるようにしたもので、図６では一例としてプロセッサＣＰＵを２つつくったマルチプロセッサであるものとして説明する。つまり、本実施例は複数のプロセッサ要素で１つの汎用レジスタグループを専有しないように、それぞれプロセッサごとに番号が決められており、管理レジスタの実行状態フラグへ割り当てられたプロセッサの番号を登録するものである。そのため、ここではプロセッサＣＰＵ１はプログラムカウンタＰＣ１と演算器ＡＬＵ１により構成されるプロセッサ要素ＰＥ１を含み、同様にプロセッサＣＰＵ２はプロセッサ要素ＰＥ２を含むものとなる。さらに、本実施例においては、図６に示すようにデータキャッシュメモリおよび命令キャッシュメモリが上記プロセッサＣＰＵ１，ＣＰＵ２に対して共通に設けられている。これに対し、従来は上述した図２のようにそれぞれのプロセッサＣＰＵ毎に専用の内蔵キャッシュメモリが設けられ、上記それぞれのプロセッサＣＰＵ同志をアクセスさせる共有のバスを設けて、２次キャッシュメモリをプロセッサ毎に共通にアクセスしていた。このため、上記プロセッサＣＰＵ同志が別チップであっても別プロセスで同一データ領域から読まれ、それぞれの結果が異なるとき、２次キャッシュメモリに退避されるデータはどちらか一方のプロセスの結果であり、他のプロセスの結果が消されてしまう。さらに、その同一データに依存する別のプロセスを起動すると誤ったデータを読み込む可能性があるため、そのようなデータのアクセスはＯＳにより禁止されている。しかし、本実施例においては、図６に示すように上記プロセッサＣＰＵ１と上記プロセッサＣＰＵ２に対応するデータキャッシュメモリおよび命令キャシュメモリが共通に設けられ実行すべきプロセスを管理レジスタにおいてプロセッサ番号をチェックすることにより、上記プロセス間のデータ衝突をさけることができ、内蔵キャッシュメモリでデータ転送を行なうため、ＯＳによるデータ領域のチェックにかかる時間を軽減でき、高速処理を行なうということが容易に達成できる。ここで、特に限定しないが、本実施例の上記データキャシュメモリおよび上記命令キャッシュメモリは上記プロセッサ数分の命令コードを発生するため、マルチポートメモリが使用される。また、本実施例においてはそれぞれのプロセッサＣＰＵごとにフェッチ回路，デコード回路，データバス，アドレスバス等が設けられるが、プロセス切り換え回路は上記データキャッシュメモリ，上記命令キャッシュメモリと同様にプロセッサの数に関係なく共通にアクセスできるように設けられる。

【００１４】以下に本実施例のマルチプロセッサのプロセス切り換え方法について説明する。まず、実施例１と同様にしてＧｒｐ１ａ，Ｇｒｐ２ａのように汎用レジスタのグルーピングを行ない、ＯＳ上でプログラムカウンタＰＣ１，ＰＣ２のそれぞれの値ＰＣ＊をそれぞれの上記管理レジスタにそれぞれ設定するとともに実行状態のフラグデータとともに格納されたプロセッサエレメントデータＰＥ＊を格納させる。そして、起動アドレス入力で上記管理レジスタ群の管理レジスタ指定を行ない、上

記汎用レジスタグループと対応したプロセスを上記管理レジスタと対応した上記汎用レジスタグループへ転送し実行する。

【００１５】次に、プロセス切り換え時の本実施例のマルチプロセッサのデータ転送手順を説明する。本実施例では一例として、一方のプロセッサＣＰＵと対応する汎用レジスタグループでプロセス切り換えを行なう場合の処理方法について以下に説明する。ここでは、プロセッサＣＰＵ１，２をそれぞれプロセッサ要素ＰＥ１，ＰＥ２と対応させるとともに、汎用レジスタファイルを管理レジスタと同数のグループに分割するものである。そのため、プロセッサＣＰＵ１においては汎用レジスタグループＧｒｐ１ａが実行状態，プロセッサＣＰＵ２においては汎用レジスタグループＧｒｐ２ａが実行状態であり、プロセッサＣＰＵ１において汎用レジスタグループＧｒｐ１ｂにプロセス切り換えを行なうものとして以下に説明する。上記プロセッサＣＰＵ１へのプロセス切り換え命令がデータキャッシュメモリ及び命令キャッシュメモリから発生されるとともに、上記プロセッサＣＰＵ２はプロセス切り換え命令が発生されていない状態とされている。そして、上記命令キャッシュメモリはフェッチ回路１を介して上記プロセッサＣＰＵ１内部へプロセス切り換え命令をフェッチする。そして、それぞれ上記フェッチ回路１によってフェッチされた命令はそれぞれデコード回路ＤＣＲ１に転送され、上記デコード回路ＤＣＲ１から出力されたデータはプロセス切り換え回路に入力される。一方、フェッチ回路２には、上記命令キャッシュメモリから上記プロセッサＣＰＵ２におけるプロセス切り換え命令が発生されていない。このため、デコード回路ＤＣＲ２を介してプロセッサ要素ＰＥ２と対応する演算器ＡＬＵ２において指定された演算内容を実行するための命令が保持され、プロセス切り換え回路にはデータが入力されず、命令実行中のプロセッサ要素ＰＥ２および管理レジスタ２ａにおけるプロセスが保持されて実行される。このようにして、上記デコード回路ＤＣＲ１のようにプロセス切り換え命令が解読されたデコード回路からのみプロセス切り換え回路への命令が取り込まれる。

【００１６】次に、このプロセス切り換え回路における処理方法について以下に説明する。まず、退避ブロックにおいては、切り換え制御線を介してレジスタ制御線とアクセスし、実行中の管理レジスタ１ａを選択する。そして、プロセッサ要素ＰＥ１におけるプログラムカウンタＰＣ１に上記プログラムカウンタの値ＰＣ＊を保持し、さらに入力用データバス１を介して上記プログラムカウンタの値ＰＣ＊を上記管理レジスタ１ａに上書きする。そして、上記管理レジスタ１ａに格納された実行状態フラグを上記レジスタ制御線を介して上記切り換え制御線上に読み出し、上記実行状態フラグおよびそこに保持されたプロセッサエレメントデータＰＥ＊としてのプロセッサ要素ＰＥ１のデータを待機状態に切り換える。次に選択アルゴリズムブロックにおいては、実行待ち状態フラグから次実行プロセスを選択し、そのプロセスと対応した汎用レジスタグループを指定するために、レジスタグループ信号Ｇｒｐ＃を発生する。そして、ラッチ回路にて上記選択アルゴリズムブロックから発生したレジスタグループ信号Ｇｒｐ＃を保持しつつ、汎用レジスタグループＧｒｐ１ｂを選択し、このことにより上記管理レジスタ１ｂが指定される。さらに、復帰ブロックにおいて、上記切り換え制御線を介して上記管理レジスタ１ｂを指定し、この管理レジスタ１ｂに格納されたプログラムカウンタＰＣ１の値ＰＣ＊を上記プログラムカウンタＰＣ１にセットする。そして、上記選択アルゴリズムブロックにてビット番号により読み出されたレジスタグループ番号から上記管理レジスタ１ｂを割り出し、上記管理レジスタ１ｂに格納された待機状態のフラグを実行状態フラグ，プロセッサ要素ＰＥ１の実行中にそれと同一ビットに切り換える。また、上記管理レジスタ１ｂ及び上記汎用レジスタグループＧｒｐ１ｂから読み出されたデータは演算器ＡＬＵ１によって演算が行われる。そして、実行再開命令が上記命令キャッシュメモリより発生されることによって、プロセス切り換え動作が終了し、プロセスの実行が行われる。ここで、ＯＳ上にデータがあるときは、上記管理レジスタ及び上記汎用レジスタグループのうちで使用頻度の少ないデータをプログラム制御あるいは自動制御により上記データキャッシュメモリあるいは上記命令キャッシュメモリへ退避し、そのキャッシュメモリから管理レジスタ群へ必要なデータを転送する。この切り換え方法はソフトウェアとハードウェア両方による実現が可能であるが、ソフトウェアだと切り換えプロセスが必要となる。また、従来は１つのプロセス汎用レジスタファイルを占有することによって数サイクルが消費されていたので、本方式を用いて高速化を図るためにはハードウェアで組み込む方が望ましい。しかし、この場合、汎用レジスタファイル自身に分割のためのハードウェアを付加するものではなくプロセス切り換えを行うものである。ここで、本実施例では演算器としてＡＬＵを記載したが、かわりにグラフィックアクセラレータを使用する、あるいは上記ＡＬＵとグラフィックアクセラレータを並列して使用することによりさらにソフトウェアの負担が軽減できるとともに、１サイクルで命令実行でき、高性能な演算処理が可能となる。グラフィックアクセラレータの場合は、グラフィック命令をそれぞれプロセスとみなすことにより、通常のＣＰＵでのプロセスと異なり、プロセス毎の依存性がないことから高速処理が可能である。また、本実施例ではプロセッサエレメントデータＰＥ＊が実行中の各管理レジスタに格納されているために、他のプロセッサ要素を共有することもできる。

【００１７】（実施例４）図７に本発明を応用して、１

つのプロセッサに対して複数のプログラムカウンタを設け、それを１サイクル単位の高速プロセス切り換えに対応できるようにしたプロセッサを示す。従来、ＲＩＳＣプロセッサのような１サイクルごとにパイプライン処理を行うプロセッサに対して、条件分岐命令による評価遅延や、メモリのロードストアによる遅延が原因となり、パイプラインに入らない命令は大抵ディレイスロットとして実行されずに１サイクルが消費されるという問題点があった。本実施例はこれを低減するための方式であり、プログラムカウンタＰＣ１，ＰＣ２のように、１つのプロセッサに対して複数のプログラムカウンタを設け、実行中のプロセスおよび非実行中のプロセスを示すアドレスを保持し、サイクル単位でプログラムカウンタを切り換えるとともに、命令コードを使いわけるものである。そして、実行中のプロセス，非実行中のプロセスを切り換えるプログラムカウンタＰＣ１，ＰＣ２のうちどちらかを選択するためのセレクタを設けるとともに、上記プログラムカウンタＰＣ１，ＰＣ２のそれぞれに対応する汎用レジスタグループと対応した管理レジスタにおけるプログラムカウンタの値ＰＣ＊を選択するＰＣ＊セレクタを設ける。そして、上記プログラムカウンタＰＣ１，ＰＣ２へのプログラムカウンタの値ＰＣ＊のセット，上記管理レジスタおよび上記汎用レジスタグループの制御を行なうものである。さらに、外部からの命令としてディレイスロットが入力されたときに、その命令を検出するためのディレイスロット検出回路が上記命令キャッシュメモリと上記フェッチ回路との間に設けられる。そして、上記ディレイスロット検出回路はディレイスロットを検出したときに上記セレクタを動作させる必要があるため、上記ディレイスロット検出回路と上記セレクタとは接続される。また、上記実施例におけるパイプラインステージ用のラッチ回路はＰＣ１用，ＰＣ２用にそれぞれ分けることも可能である。

【００１８】ここで、図８（ａ）は命令キャッシュメモリに格納された命令内容を示すものであり、図８（ｂ）は命令実行のダイアグラムを示すものである。この図８（ａ），図８（ｂ）について以下に説明する。図８（ａ）でプログラムカウンタＰＣ１で実行されるデータの中でＤＳ１とＤＳ２はディレイスロットであり、なにもしないということを実行する同じ命令コードが入っている。また、プログラムカウンタＰＣ２は実行待ちのプロセスの中から選択されたものであり、いつ実行されても良い実行状態プロセスであり、特に限定しないが、ＰＣ１のプロセス終了後、ＰＣ２がメインの実行プロセスとなる。命令をフェッチする前に上記プログラムカウンタＰＣ１の示すアドレスが上記ディレイスロットＤＳ１を示したときに、ディレイスロットの命令を検出してセレクタを上記プログラムカウンタＰＣ２に切り換え、ディレイスロットではない命令コードをフェッチデータとして読み込む。同様にしてディレイスロットＤＳ２も上記プログラムカウンタＰＣ２で実行する命令コードをフェッチデータとして読み込む。また、上記プログラムカウンタＰＣ１の命令コードを実行するときに、演算器ＡＬＵは汎用レジスタグループの間で演算を行い、上記プログラムカウンタＰＣ２の命令コードを実行するときには、汎用レジスタグループの間で演算を行う。このようにして、上記プログラムカウンタＰＣ１をパイプラインに命令を埋める補助プログラムとして使用することによって１プロセッサに対し、１サイクルあたり１命令の実行が可能となる。しかしながら、上記プログラムカウンタＰＣ２の命令コードにも例外なくディレイスロットが入っているため、上記プログラムカウンタＰＣ２もディレイスロットの検出が必要である。これは、上記プログラムカウンタＰＣ１の実行中にディレイスロットのサイクルを検出し、消費すれば良いのでフェッチサイクルの中にディレイスロットが入ることはほとんどない。

【００１９】次に、図７，図８（ａ），図８（ｂ）を用いて本実施例におけるプロセッサのプログラムカウンタの切り換え方法についてのデータ転送手順を説明する。また、プロセス切り換え動作は実施例１と同様のため、説明を省略する。上記プログラムカウンタＰＣ１が図８（ａ）におけるＰＣ１のディレイスロットＤＳ１からオペランドＯＰ２２へ命令を切り換えるものとして以下に説明する。まず、なにもしないということを実行するＮＯＰ命令が命令キャッシュメモリから出力される。そして、上記命令キャッシュメモリを介してＮＯＰ命令がディレイスロットＤＳ検出回路に読み込まれ、上記命令キャッシュメモリからの命令がディレイスロットＤＳ１であることを検出する。そして、セレクタへプロセス切り換えを指定し、プログラムカウンタを切り換えるためのデータを転送し、実行中のプログラムカウンタＰＣ１をＰＣ２に切り換える。そして、それぞれ上記フェッチ回路によってフェッチされた命令はそれぞれデコード回路ＤＣＲに転送され、命令解読が行われるとともに演算器ＡＬＵの演算内容を選択し、上記演算器ＡＬＵへその演算内容データを転送する。さらに、上記デコード回路ＤＣＲから出力されたデータは管理レジスタ並びに汎用レジスタに取り込まれ、管理レジスタ２ａにてＯＰ２２を実行し、次の命令を命令キャッシュメモリから発生することにより、上記プログラムカウンタＰＣ１に切り換えられる。

【００２０】ここで、本実施例では演算器としてＡＬＵを記載したが、かわりにグラフィックアクセラレータを使用する、あるいは上記ＡＬＵとグラフィックアクセラレータを並列して使用することによりさらにソフトウェアの負担が軽減できるとともに、高性能な処理が可能となる。本実施例は、汎用レジスタグループの数が少ないときや特殊レジスタとして扱われる場合は、汎用レジスタとそれに対応するプログラムカウンタとの対応がとれていれば管理レジスタは必ずしも必要とされないもので

あって、プログラムカウンタは2つ以上設定することもできる。そして、必ず1つのプログラムカウンタを実行用に使用し、その他のプログラムカウンタはディレイスロット置き換え用に使用する。

【0021】（実施例5）図9に本発明のマイクロプロセッサを適用したワークステーションの機能ブロック図を示す。システムバスにより、本発明を適用した複数のプロセッサCPU，メモリマネージメントユニットMMU，メインメモリ，グラフィックアクセラレータ，入出力装置I／O，2ndキャッシュメモリが接続されている。そして、上記グラフィックアクセラレータとCRTが接続され、上記入出力装置I／Oとネットワーク，キーボード，ハードディスクとが接続されている。ここで、上記メモリマネージメントユニットMMUは、階層化したメモリと実メモリとを変換するためのものであって、上記グラフィックアクセラレータは点，線，塗り潰し，文字等の高速描画を命令により読み込み実行するものであり、上記グラフィックアクセラレータによる演算結果はVRAMへドットデータ出力として書き込み、CRTに出力されて表示される。そして、上記入出力装置I／Oは基本的に本発明のプロセッサCPUからの命令にしたがってワークステーション外部とのデータの入出力を行なうが、DMA転送の際にはメモリーI／O間のデータ転送を行なう。さらに、上記入出力装置I／Oは上記キーボードから入力された外部からの命令を取り込むとともに、上記ネットワークにより他のワークステーション上との通信を行なう。そして、上記ハードディスクは本発明のプロセッサから発生された図示していない上記ハードディスクのハードディスクドライバに対してこのハードディスクの内容を取り出す量とそのメインメモリへの格納場所を指定する。ここで、本発明のプロセッサは1つの1次キャッシュメモリを複数のプロセッサ要素PE1～PEnで共用できるために、従来ソフトウェアで切り換えを行なっていたマルチスレッドの一部をハードウエアで切り換えることが可能となるので、メモリアクセス回数が低減でき、高速切り換えが可能となる。また、従来は内容を認識するまで他のプロセッサCPUを動作させることが不可能であったが、内部バスで対応させることが可能となるので、プロセッサCPU内で演算可能となるためにスケジューリングが高速に行なえる。さらに、外部割込みの時の実行プロセスを管理レジスタの中に格納すると、割込み要求後のプロセス切り換えを高速化することができる。

【0022】

【発明の効果】大容量の汎用レジスタファイルをもつ論理集積回路において、汎用レジスタファイルをプロセスごとに分割することにより、プロセス切り換えの時間を大幅に短縮し、マルチプロセッサへの拡張を容易にし、パイプライン処理に割り当てる命令コードの最適化により、処理速度を大幅に短縮することができる。

【図面の簡単な説明】

【図1】本発明を適用したプロセッサの機能ブロック図。

【図2】従来のRISCプロセッサの機能ブロック図。

【図3】一般的なRISCプロセッサのパイプライン処理の概念図。

【図4】本発明のプロセッサにおいて実行状態フラグを読み書きするための方法を示す図。

【図5】実施例2の管理レジスタに汎用レジスタグループのデータを格納したプロセッサの機能ブロック図。

【図6】実施例3の複数の演算器と複数のプログラムカウンタを設けたマルチプロセッサの機能ブロック図。

【図7】実施例4の1つのプロセッサに対して複数のプログラムカウンタを設けたプロセッサの機能ブロック図。

【図8】実施例4の1つのプロセッサに対して複数のプログラムカウンタを設けたプロセッサの命令キャッシュメモリの命令内容とパイプライン処理を示す図。

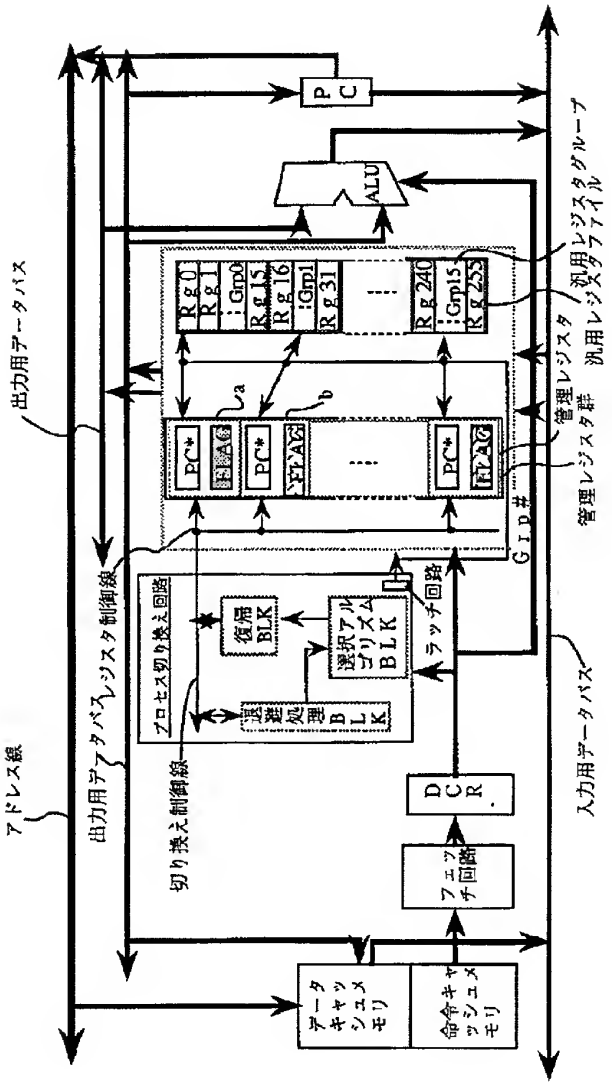
【図9】実施例5の本発明を適用したワークステーションシステムの機能ブロック図。

【符号の説明】

PC…プログラムカウンタ，Reg…汎用レジスタ，DCR…デコード回路，BLK…ブロック，Grp…汎用レジスタグループ，PC＊…プログラムカウンタの値，ALU…演算器，Grp＃…レジスタグループ信号，CPU…プロセッサ，grp＊…汎用レジスタグループデータ，PE＊…プロセッサエレメントデータ，PE…プロセッサ要素，MMU…メモリマネージメントユニット，I／O…入出力装置。

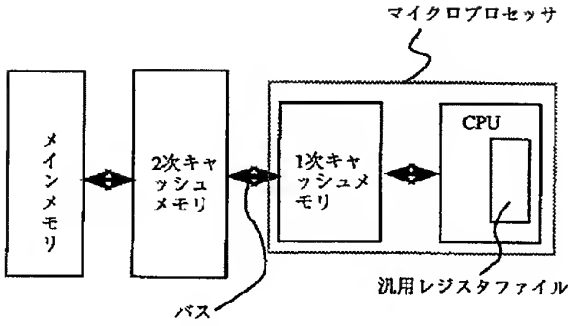
【図1】

【　図　1　】


PC
ALU
Rg0
Rg1
:Grp0
Rg15
Rg16
:Grp1
Rg31
Rg240
:Grp15
Rg255
汎用レジスタグループ
汎用レジスタファイル
管理レジスタ群
管理レジスタ
a
b
PC*
EPA0
EPA1
EPA15
GID#
出力用データバス
レジスタ制御線
プロセス切り換え回路
復帰BLK
選択アルゴリズムBLK
退避処理BLK
ラッチ回路
入力用データバス
アドレス線
出力用データバス
切り換え制御線
DCR
フェッチ回路
データキャッシュメモリ
命令キャッシュメモリ

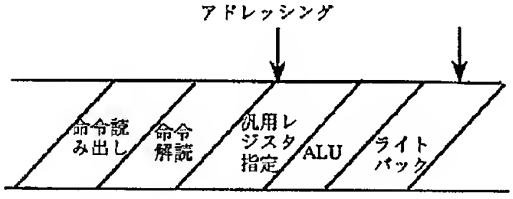
【図２】

【 図 2 】



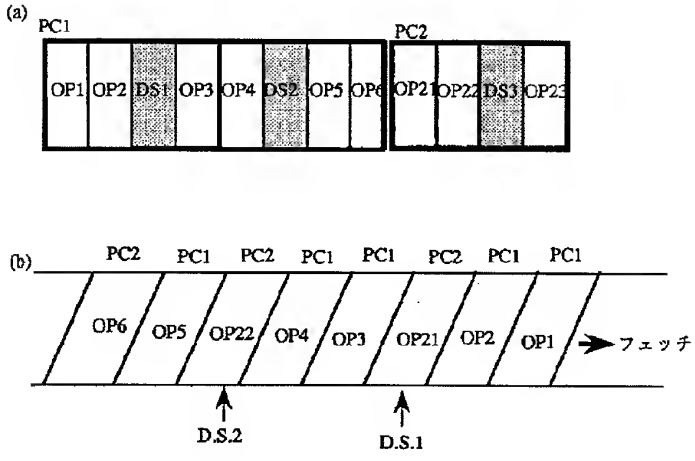
【図３】

【 図 3 】



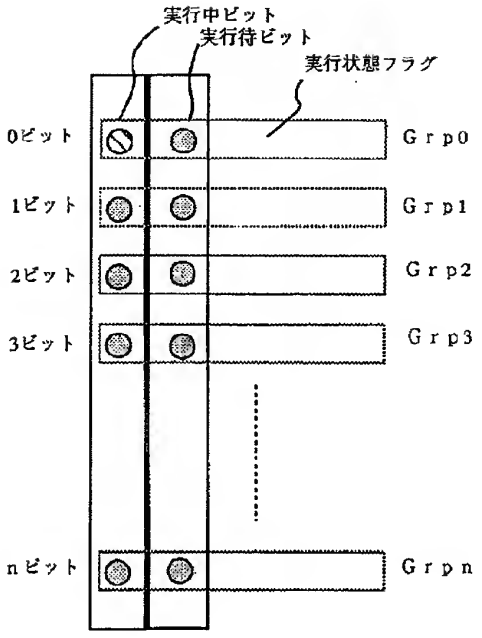
【図８】

【 図 8 】



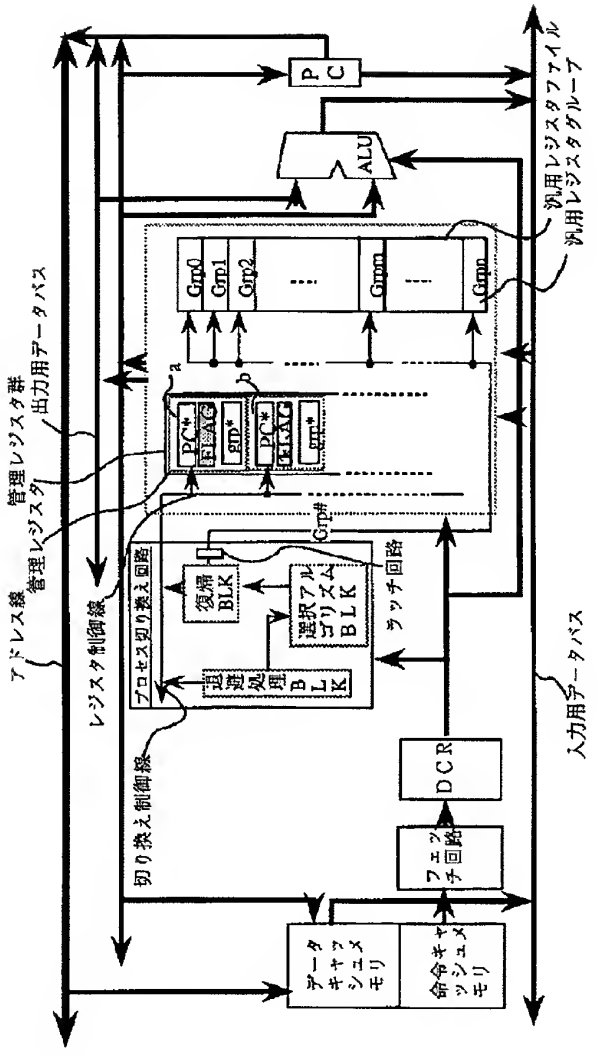
【図4】

【 図 4 】


実行中ビット
実行待ビット
実行状態フラグ
0ビット
1ビット
2ビット
3ビット
nビット
Grp0
Grp1
Grp2
Grp3
Grpn

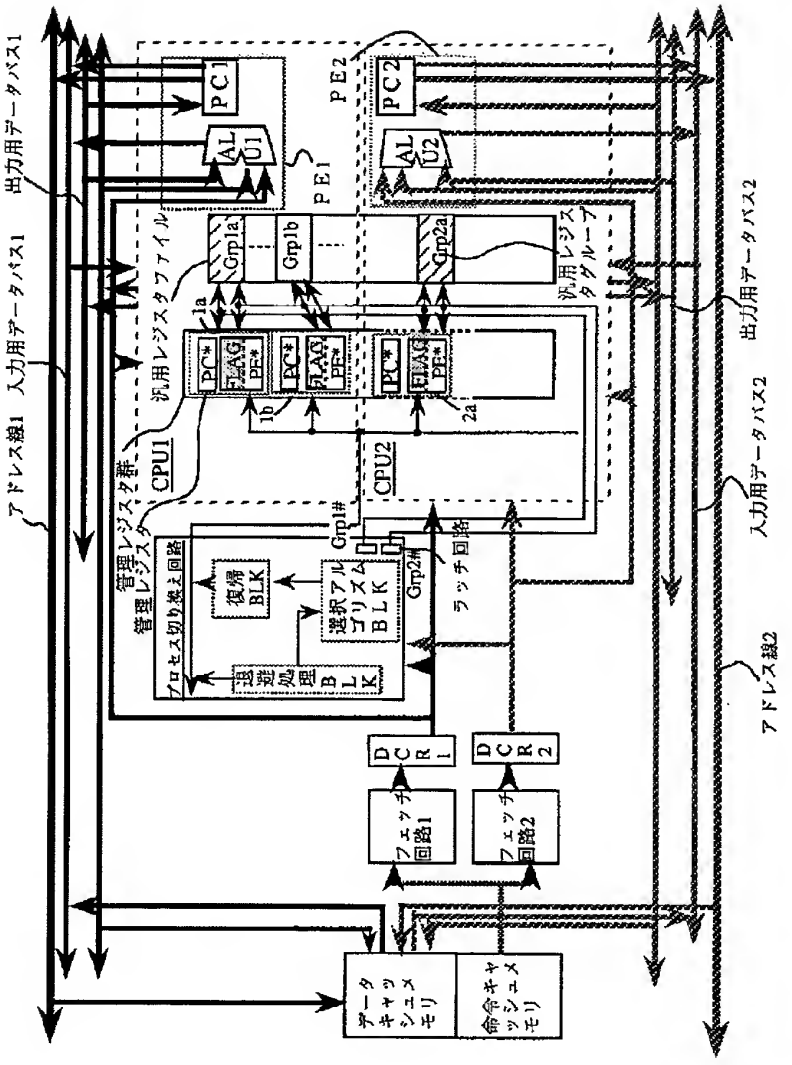
【図5】

【　図　5　】


P C
ALU
汎用レジスタファイル
汎用レジスタグループ
Grp0
Grp1
Grp2
Grpm
Grpn
出力用データバス
管理レジスタ群
管理レジスタ
PC*
FLAG
grp*
アドレス線
レジスタ制御線
プロセス切り換え回路
復帰BLK
選択アルゴリズムBLK
退避処理BLK
ラッチ回路
切り換え制御線
DCR
フェッチ回路
データキャッシュメモリ
命令キャッシュメモリ
入力用データバス

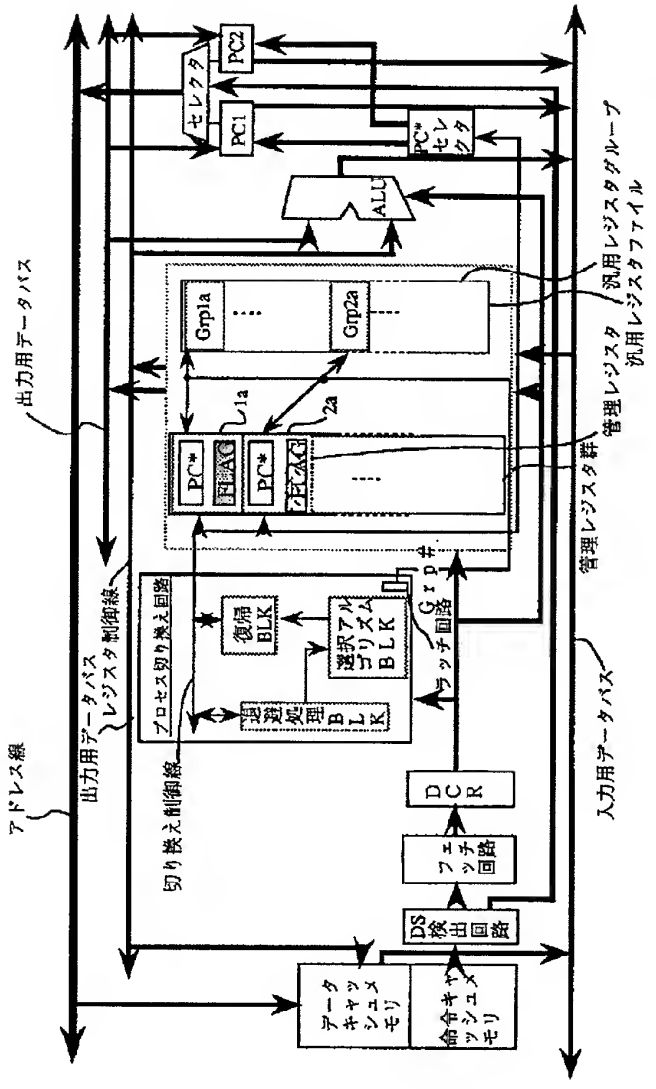
【図6】

【 図 6 】

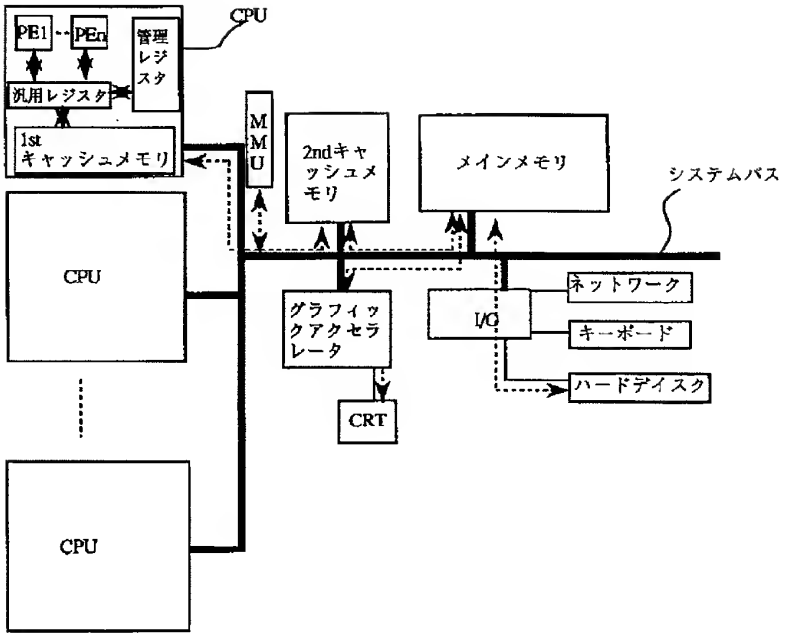

アドレス線1
入力用データバス1
出力用データバス1
PE1
PE2
PC1
PC2
ALU1
ALU2
Grp1a
Grp1b
Grp2a
汎用レジスタファイル
汎用レジスタグループ
PC*
TAG PE*
CPU1
CPU2
管理レジスタ群
管理レジスタ
プロセス切り換え回路
復帰BLK
選択アルゴリズムBLK
退避処理BLK
Grp1#
Grp2#
ラッチ回路
DCR1
DCR2
フェッチ回路1
フェッチ回路2
データキャッシュメモリ
命令キャッシュメモリ
出力用データバス2
入力用データバス2
アドレス線2

【図7】

【　図　7　】


出力用データバス
アドレス線
出力用データバス
レジスタ制御線
切り換え制御線
プロセス切り換え回路
復帰BLK
選択アルゴリズムBLK
例外処理BLK
Grp#
ラッチ回路
DCR
フェッチ回路
DS検出回路
データキャッシュメモリ
命令キャッシュメモリ
入力用データバス
管理レジスタ群
管理レジスタ
汎用レジスタグループ
汎用レジスタファイル
PC2
セレクタ
PC1
PC*セレクタ
ALU
Grp1a
Grp2a
1a
2a
PC*
FLAG
PC*
FLAG

【図９】


PE1
PEn
管理レジスタ
汎用レジスタ
1st キャッシュメモリ
CPU
MMU
2ndキャッシュメモリ
メインメモリ
システムバス
CPU
グラフィックアクセラレータ
I/O
ネットワーク
キーボード
ハードディスク
CRT
CPU

【図９】